* What are Monte Carlo approaches
* How are they potentially better than the policy and value iteration algorithms described so far

Monte Carlo approach, is where you run a system a significant number of episodes with quasi-random inputs. The desired outputs are then measured, analysed or learned from directly, by the agent. It improves the policies and values every episode, thus it can only be applied to episodic tasks that eventually terminate, as its learning approach is based off averaging sample returns. In policy iteration, we can use Monte Carlo techniques to both, evaluate a policy and also improve a policy based off empirical simulations.

Monte Carlo approaches however, are computationally slow and expensive. The mean has a convergence of and convergence of covariance is . Therefore, for error to be zero, would need to be infinite but since we always have finite , Monte Carlo estimates are never exact. However typically we work with generalised policy iterations which can work with inexact values.

It can be better than policy and value iteration algorithms because of its real-time learning characteristics. It uses model-free learning, meaning it does not need to have complete prior knowledge of the environment. Furthermore, it can be more efficient to run because it does not need to travel to every single state and so, when calculating policies, will only need to update the states that it has visited which tends to be the most common/important. Unlike policy and value iteration which are model-based and need to sample over an entire state space to specify a policy.

**New Section:**

Monte Carlo approaches

* Basic rundown of how monte carlo works
  + Run system lots of times
  + Work out expected returns using empirical data
* Monte carlo can do both the improve and evaluation steps
* Why is monte carlo better
  + Doesn’t require a model
  + Doesn’t require sampling over entire state space, policy can be restricted to mc policy run data
  + Supports real-time learning

Monte Carlo approaches involve running a system through many times with quasi-random inputs to generate a lot of episodes. The desired outputs from these episodes are then measured, analysed, and learned from directly by the agent, with the policies and values improving over time. The learning period is then terminated when the policy converges or a specified number of episodes is reached, with more episodes in general, leading to a more optimal policy. The agent can then use this learned information to accomplish a task, such as how to navigate an airport. However, as its learning approach is based off averaging sample returns, applications are limited to episodic tasks that eventually terminate, as a policy has to be extracted. Similar to policy iteration, we can use Monte Carlo techniques to both, evaluate a policy and also improve a policy based off empirical simulations.

Due to Monte Carlo using empirical data to determine the best policy, the results obtained will always have a level of uncertainty associated with them and will never be exact. This is not a major issue as typically work is done with generalised policy iterations which works with inexact values. However, when sampling a variable, mean has a convergence of and convergence of covariance is , where is the number of times the variable is measured. This means that Monte Carlo is a computationally slow

This trade-off in computational speed does provide a set of benefits which make Monte Carlo better and more robust than the policy and value iteration methods discussed earlier in the module. The best benefit of Monte Carlo algorithms is model-free policy iteration. Previous approaches seen requires a state transition model to find the optimal value function and policy.

Where is the optimal value function and is the state transition probability distribution. With the empirical approach that Monte Carlo uses this can be estimated by directly using the expected value of the return which achieves the optimal policy allowing the optimal value function. The expected value of the return incorporates future effects and allows Monte Carlo to not need any prior knowledge of the environment such as pre-defined models of the state transitions or reward, instead learning directly from the data.

Additionally, although the final policy will still need to be extracted, real-time evaluation can be accomplished by feeding data directly to the Monte Carlo algorithm as events occur, this is not possible in policy or value iteration due to the need for prior knowledge about the environment. Unlike policy and value iteration, the Monte Carlo approach also doesn’t need the entire state space to be sampled to specify policies as the calculations can be restricted to just states with data. This allows Monte Carlo to be more efficient as the algorithm does not need to iterate across all the states and can restrict data generation and gathering to areas of interest.